Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2018)

VHSIC Hardware Description Language

Breve Tutorial de VHDL para Vivado

1. Descripción

VHDL es un lenguaje que nos permite simular el comportamiento de circuitos. A diferencia de un lenguaje tradicional de programación, que se ejecuta secuencialmente, lo que es descrito en el lenguaje se emula en circuito concurrente. Originalmente fue desarrollado por el departamento de defensa de los Estados Unidos para simular circuitos, pero hoy también se utiliza para programar FPGAs como la Basys3.

2. Crear Componentes

Para iniciarlos en el proyecto partiremos creando un Half Adder. Primero deben agregar un nuevo diseño desde "Proyect Manager - Add Source" y elegiremos "Add or create design sources". Desde esa opción crearemos un nuevo archivo de tipo VHDL y nombre "HA", se pueden agregar o importar mas de un archivo de ser necesario. Una vez finalizado nos permitirá' elegir el nombre y las características de los puertos de entrada y salida de los componentes que crearon, en este caso "a" y "b" como entradas, y "s" y "c" como salidas.

```
1  library IEEE; -- includes
2  use IEEE.STD_LOGIC_1164.ALL;
3
4
5  entity HA is -- declaracion de la entidad
6      Port ( a : in  std_logic ; -- definicion de puertos
7             b : in  std_logic ;
8             s : out std_logic ;
9             c : out std_logic );
10 end HA;
11
12 architecture Behavioral of HA is
13         -- declaraciones y definiciones
14 begin
15         -- conecciones e instancias
16 end Behavioral;
```

Código generado con una breve descripción de las partes.

Como vieron en clases el comportamiento de un HA debe simular la suma dos bits “a” y “b”, dando como resultado una suma “s” y un carry “c”. Esto implica que “s” será 1 solo si “a” o “b” es 1, y “c” será 1 solo si “a” y “b” son 1. Las compuertas lógicas para esas operaciones son Xor y And respectivamente. Para expresar eso el código es el siguiente:

```
12 architecture Behavioral of HA is
13
14 begin
15
16 s <= a xor b;
17 c <= a and b;
18
19 end Behavioral;
```

3. Usar Componentes

Para anidar componentes primero deben ser ser declarados, luego instanciados y conectados. a continuación un ejemplo de como declarar e instanciar dos HA.

```
12 architecture Behavioral of FA is
13
14 component HA -- Declaracion del HA
15     Port ( a  : in  std_logic;
16            b  : in  std_logic;
17            s  : out std_logic;
18            c  : out std_logic);
19     end component;
20
21 signal s1 :  std_logic;  -- Declaracion de senales
22 signal c1 :  std_logic;
23 signal c2 :  std_logic;
24
25 begin
26
27 c <= c1 and c2;
28
29 inst_HA: HA port map( -- Primera instancia de HA
30          a       =>'1', -- Conexiones de los puertos de la instancia
31          b       =>'0',
32          s       =>s1,
33          c       =>c1
34      );
35
36 inst_HA2: HA port map( -- Segunda instancia de HA
37          a       =>'0',
38          b       =>'1',
39          s       =>'0',
40          c       =>c2
41      );
42
43 end Behavioral;
```

No usar como referencia, la lógica de las conexiones es incorrecta.

4. Bonus: Muxers

El código a continuación representa 4 muxers, impleméntelos en su proyecto para probarlos.

```
42 begin
43
44 with sw select
45     dis_a <= "0111" when "110",
46     "0000" when others;
47
48 with sw select
49     dis_b <= "0111" when "110",
50     "0000" when others;
51
52 with sw select
53     dis_c <= "0001" when "110",
54     "0000" when others;
55
56 with sw select
57     dis_d <= "1000" when "110",
58     "0000" when others;
```